CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2013; **25**:1722–1737
Published online 18 January 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.2979

SPECIAL ISSUE PAPER

A scalable Helmholtz solver in GRAPES over large-scale multicore cluster

Linfeng Li [1], Wei Xue [1,*,†], Rajiv Ranjan [2] and Zhiyan Jin [3]

[1] *Department of Computer Science and Technology, Tsinghua University, Beijing, China*
[2] *CSIRO ICT Centre, Canberra, Australia*
[3] *National Meteorological Center, Beijing, China*

SUMMARY

This paper discusses performance optimization on the dynamical core of global numerical weather prediction model in Global/Regional Assimilation and Prediction System (GRAPES). GRAPES is a new generation of numerical weather prediction system developed and currently used by Chinese Meteorology Administration. The computational performance of the dynamical core in GRAPES relies on the efficient solution of three-dimensional Helmholtz equations, which lead to large-scale and sparse linear systems formulated by the discretization in space and time. We choose generalized conjugate residual (GCR) algorithm to solve the corresponding linear systems and further propose algorithm optimizations for large-scale parallelism in two aspects: (i) reduction of iteration number for solution and (ii) performance enhancement of each GCR iteration. The reduction of iteration number is achieved by advanced preconditioning techniques, combining block incomplete LU factorization-k preconditioner over 7-diagonals of the coefficient matrix with the restricted additive Schwarz method effectively . The improvement for GCR iteration is to reduce the global communication operations by refactoring the GCR algorithm, which decreases the communication overhead over large number of cores. Performance evaluation on the Tianhe-1A system shows that the new preconditioning techniques reduce almost one-third iterations for solving the linear systems, the proposed methods can obtain 25% performance improvement on average compared with the original version of Helmholtz solver in GRAPES, and the speedup with our algorithms can reach 10 using 2048 cores compared with 256 cores. Copyright © 2013 John Wiley & Sons, Ltd.

Received 4 June 2012; Revised 29 October 2012; Accepted 31 October 2012

KEY WORDS: numerical weather prediction; Helmholtz equation; ILU; additive Schwarz method; improved GCR algorithm

1. INTRODUCTION

1.1. Introduction to GRAPES

Weather forecast has an important impact on agriculture and industry as well as people's daily life. With the progress of theory for numerical weather prediction (NWP) and the rapid development of high-performance computing, NWP is becoming the most important way and the only quantitative way for future weather prediction.

Global/Regional Assimilation and Prediction System (GRAPES) was developed since 2001 and is one of the operation models widely used in China [1]. To improve the forecast accuracy, developing high-resolution model is considered as the most effective way in GRAPES as well as in all of the well-known NWPs. The owner of GRAPES, Chinese Meteorology Administration, has

*Correspondence to: Wei Xue, Department of Computer Science and Technology, Tsinghua University, Beijing, China, 100084.
†E-mail: xuewei717@gmail.com

used the GRAPES global model with 25-km horizontal resolution for pre-operation by 2011, which needs many computing resources to meet the requirement for on-line operation. Finer resolution global model (such as 15-km model) has already been under development. So the computation performance is becoming a big challenge for GRAPES. In the dynamical core of GRAPES, the semi-implicit semi-Lagrangian time-stepping scheme requires the solution of global three-dimensional (3D) Helmholtz equations. And the solution of the Helmholtz equations is finally transformed into solving very large-scale sparse linear equations for each time step, for example, the dimension of the coefficient matrix in a 25-km resolution model is 37,324,800. The time consumed for solving the equations is the computational bottleneck of high-resolution global model in GRAPES, which limits the overall computational performance.

In this paper, we choose generalized conjugate residual (GCR) algorithm for solving the linear systems in GRAPES and give a detail comparison between GCR and other well-known iterative methods including generalized minimal residual algorithm (GMRES), biconjugate gradient stabilized method (BiCGSTAB), and induced dimension reduction method (IDR). Algorithm optimizations are further proposed for large-scale parallelism in two aspects: (i) reduction of iteration number for solution and (ii) performance enhancement of each GCR iteration. The reduction of iteration number is achieved by advanced preconditioning techniques, combining block incomplete LU factorization with level 2 fill (ILU(2)) preconditioner over 7-diagonals of the coefficient matrix with the restricted additive Schwarz (RAS) method effectively. The improvement for GCR iteration is to reduce the number of global communication operations to only one in each iteration by refactoring the GCR algorithm, which decreases the communication overhead over large number of cores largely. Performance evaluation on the Tianhe-1A system shows that the new preconditioning technique reduces almost one-third iterations for solving the linear systems, the proposed methods can obtain 25% performance improvement on average compared with the original version of Helmholtz solver in GRAPES [2], and the speedups with our algorithms can reach 10 using 2048 cores compared with 256 cores on the Tianhe-1A system.

1.2. Helmholtz equation in GRAPES

With the discretization of space and time, we can obtain the following prognostic equations in GRAPES global model [3].

$$u^{n+1} = \left[\epsilon_{u1}\frac{1}{\alpha\cos\varphi}\frac{\partial}{\partial\lambda} + \epsilon_{u2}\frac{1}{\alpha}\frac{\partial}{\partial\varphi} + \epsilon_{u3}\frac{\partial}{\partial z}\right](\Pi')^{n+1} + \epsilon_{u0} \tag{1}$$

$$v^{n+1} = \left[\epsilon_{v1}\frac{1}{\alpha\cos\varphi}\frac{\partial}{\partial\lambda} + \epsilon_{v2}\frac{1}{\alpha}\frac{\partial}{\partial\varphi} + \epsilon_{v3}\frac{\partial}{\partial z}\right](\Pi')^{n+1} + \epsilon_{v0} \tag{2}$$

$$w^{n+1} = \left[\epsilon_{w1}\frac{1}{\alpha\cos\varphi}\frac{\partial}{\partial\lambda} + \epsilon_{w2}\frac{1}{\alpha}\frac{\partial}{\partial\varphi} + \epsilon_{w3}\frac{\partial}{\partial z}\right](\Pi')^{n+1} + \epsilon_{w0} \tag{3}$$

$$(\Theta')^{n+1} = \left[\epsilon_{\theta1}\frac{1}{\alpha\cos\varphi}\frac{\partial}{\partial\lambda} + \epsilon_{\theta2}\frac{1}{\alpha}\frac{\partial}{\partial\varphi} + \epsilon_{\theta3}\frac{\partial}{\partial z}\right](\Pi')^{n+1} + \epsilon_{\theta0} \tag{4}$$

$$\begin{aligned}(\Pi')^{n+1} = \Bigg[&\frac{\epsilon_{\Pi1}}{\alpha\cos\varphi}\frac{\partial}{\partial\lambda} + \frac{\epsilon_{\Pi2}}{\alpha}\frac{\partial}{\partial\varphi} + \epsilon_{\Pi3}\frac{\partial}{\partial z} + \frac{\epsilon_{\Pi_{H1}}}{(\alpha\cos\varphi)^2}\frac{\partial^2}{\partial^2\lambda} + \frac{\epsilon_{\Pi_{H2}}}{\alpha^2}\frac{\partial^2}{\partial^2\varphi} + \epsilon_{\Pi_{H3}}\frac{\partial^2}{\partial^2 z}\\ &+ \frac{\epsilon_{\Pi_{H4}}}{\alpha^2\cos\varphi}\frac{\partial^2}{\partial\varphi\partial\lambda} + \frac{\epsilon_{\Pi_{H5}}}{\alpha\cos\varphi}\frac{\partial^2}{\partial\lambda\partial z} + \frac{\epsilon_{\Pi_{H6}}}{\alpha}\frac{\partial^2}{\partial\varphi\partial z}\Bigg](\Pi')^{n+1} + \epsilon_{\Pi0}\end{aligned} \tag{5}$$

In which, u, v, w, Θ, and Π' are the prognostic variables in GRAPES. And ϵ_{u1}, ϵ_{u2}, ϵ_{u3}, ϵ_{v1}, ϵ_{v2}, ϵ_{v3}, ϵ_{w1}, ϵ_{w2}, ϵ_{w3}, $\epsilon_{\theta1}$, $\epsilon_{\theta2}$, $\epsilon_{\theta3}$, $\epsilon_{\Pi1}$, $\epsilon_{\Pi2}$, $\epsilon_{\Pi3}$, $\epsilon_{\Pi_{H1}}$, $\epsilon_{\Pi_{H2}}$, $\epsilon_{\Pi_{H3}}$, $\epsilon_{\Pi_{H4}}$, $\epsilon_{\Pi_{H5}}$, and $\epsilon_{\Pi_{H6}}$ are the coefficients of equations, which do not change with time. Whereas ϵ_{u0}, ϵ_{v0}, ϵ_{w0}, $\epsilon_{\theta0}$, and $\epsilon_{\Pi0}$ are

DOI: 10.1002/cpe

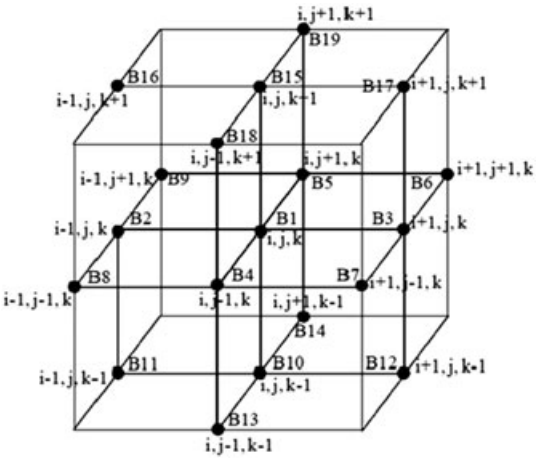

Figure 1. The pattern of 19 nonzeros in three-dimensional Helmholtz equation of Global/Regional Assimilation and Prediction System.

the coefficients change with time. φ and λ are the latitude and longitude in spherical coordinates, z is the terrain-following coordinate, and α is the radius of the Earth.

For these prognostic equations earlier, Equation (5) is the typical Helmholtz equations. The solution of Equation (5) refers to the perturbation of Exner pressure to reference atmosphere. When it is solved, Equations (1)–(4) can be calculated separately. With 3D finite difference scheme used by GRAPES, each row of the coefficient matrix of the Helmholtz equations has at most 19 nonzeros [4–6], and the pattern of these nonzeros in space can be found in Figure 1.

And the discrete formulation of grid point (i, j, k) in Equation (5) can be written as the following form:

$$\begin{aligned} &B_1(\Pi')_{i,j,k} + B_2(\Pi')_{i-1,j,k} + B_3(\Pi')_{i+1,j,k} + B_4(\Pi')_{i,j-1,k} + B_5(\Pi')_{i,j+1,k} + B_6(\Pi')_{i+1,j+1,k} \\ &+ B_7(\Pi')_{i+1,j-1,k} + B_8(\Pi')_{i-1,j-1,k} + B_9(\Pi')_{i-1,j+1,k} + B_{10}(\Pi)_{i,j,k-1} + B_{11}(\Pi')_{i-1,j,k-1} \\ &+ B_{12}(\Pi')_{i+1,j,k-1} + B_{13}(\Pi')_{i,j-1,k-1} + B_{14}(\Pi')_{i,j+1,k-1} + B_{15}(\Pi')_{i,j,k+1} + B_{16}(\Pi')_{i-1,j,k+1} \\ &+ B_{17}(\Pi')_{i+1,j,k+1} + B_{18}(\Pi')_{i,j-1,k+1} + B_{19}(\Pi')_{i,j+1,k+1} = (\xi_{\Pi 0})_{i,j,k} \end{aligned} \tag{6}$$

In which, B_i $(i = 1, ..., 19)$ is the coefficient for each grid, does not change with time, whereas $\xi_{\Pi 0}$ will change step by step.

For this kind of problem, iterative algorithms are commonly used as an effective method. In the original version of GRAPES, GCR algorithm is adopted as well as blocked ILU(0) preconditioner. Figure 2 shows the time for solving Helmholtz equation is over 35% in the computing time of the global model with different degrees of parallelism. And the Helmholtz solver is the most time-consumed kernel in GRAPES. So, improving the computing efficiency of this solver can enhance the performance of GRAPES obviously.

The rest of this paper is organized as follows. The related work is presented in Section 2. In which, we summarize the dynamical cores of known NWP systems as well as the algorithms related to solving large-scale sparse linear systems. In Section 3, we propose optimization strategies for solving the Helmholtz equations in detail, which includes the comparison between different iterative methods, the combined preconditioning techniques, and the improved GCR algorithm (IGCR) for high-resolution global model of GRAPES. Then, we show the performance results on the Tianhe-1A system in Section 4. The paper is concluded in Section 5.

2. RELATED WORK

Nowadays, a number of NWP systems have been running in different countries all over the world. Among these systems, the most popular global models are Unified Model (UM) of the UK Met office, Integrated Forecast System (IFS) from the European Center for Medium-Range

DOI: 10.1002/cpe

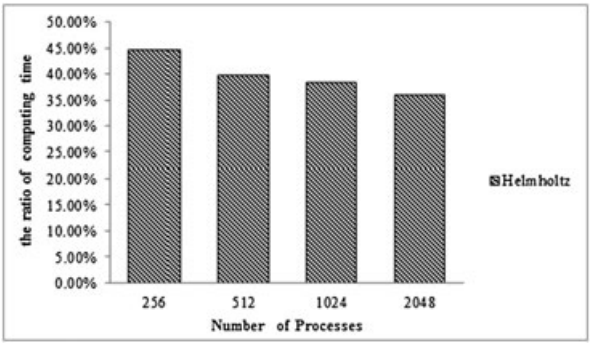

Figure 2. The time proportion for solving Helmholtz equation with different processes using the grid of 15-km horizontal resolution and 36 vertical levels.

Weather Forecast, and Global Forecast System (GFS) of the US National Centers for Environmental Prediction. The dynamical core of UM is to solve the fully nonhydrostatic Euler equations using a conservative finite difference scheme over latitude–longitude grid as well as the semi-Lagrangian advection and semi-implicit time stepping. The current horizontal resolution for operation of UM is 40 km. Different from UM, IFS uses spectral transform method, which involves the computation in different spaces, such as grid-point space, Fourier space, and spectral space. The advection schemes in IFS are Eulerian scheme and semi-Lagrangian scheme. The current horizontal resolution of IFS is around 16 km. GFS also uses the spectral method, and its time integration is leapfrog and semi-implicit method. The current horizontal resolution of GFS is around 25 km. GRAPES is widely used in China. And the dynamical core of GRAPES uses grid-based finite difference scheme for solving fully compressible, nonhydrostatic equations and adopts the semi-implicit and semi-Lagrangian time integration scheme, which is similar to UM from UK. The current horizontal resolution in GRAPES global model is 25 km, and the 15-km resolution model is under development. This paper focuses on the performance optimization over large-scale parallelism for the dynamical core in high-resolution global model of GRAPES.

In GRAPES, solving 3D Helmholtz equations is the most time-consuming parts in the dynamical core, which leads to solving large-scale sparse linear systems. Iterative methods are better for use than direct method for large-scale sparse linear system. It is because the direct method scales poorly with increasing problem size in terms of operation counts and memory requirements, especially on the problems arising from the discretization of 3D PDEs [7, 8]. Krylov subspace methods are considered as one kind of the most important iterative methods, such as GMRES and BiCGSTAB. As is well known, finding a good preconditioner is extremely important in iterative algorithms, which is viewed as a combination of art and science [9]. The term of preconditioning means to transform the system into another system with more favorable spectral properties, which is much easier to convergence [7]. ILU factorization methods, sparse approximate inverse, multigrid methods [10] are widely adopted for large-scale sparse linear systems of many scientific and engineering problems. In the previous version of GRAPES, GCR algorithm and blocked ILU(0) preconditioner are used because of acceptable performance for relative low-resolution cases with hundreds of processing cores and easy implementation [2]. Using high-resolution global model of GRAPES, it is difficult to meet the requirement for on-line operation even using much more computing resources. To further improve the computational performance of Helmholtz solver in GRAPES, we propose a novel preconditioner in this paper. Our new preconditioner combines inter-domain preconditioning with intra-domain preconditioning, which improve the convergence largely. And the inter-domain preconditioning makes use of the idea of RAS method [11, 12], and the intra-domain preconditioning uses high-level ILU-k preconditioner over 7-diagonals of the coefficient matrix for the trade-off between convergence and computation overhead.

For large-scale scientific computing, the bottleneck is usually due to inner products enforcing global communication. Many researchers make contributions to the communication issue for linear solver on distributed memory systems. Using some numerical methods and changing the order of

DOI: 10.1002/cpe

computing can reduce the number of the global communication while maintaining the numerical properties of the original methods, such as improved GCR [13], improved BiCGSTAB [14], and IDR(s) [15]. In this paper, we extend the IGCR in [13] to a restart version and make it a successful use in the linear systems with hundreds of million unknowns.

For performance tuning of NWP systems, Daniel's work [16] focuses on the architecture optimizations such as cache/memory usage and the communication optimization such as message hiding. And Alvaro's work [17] tried to solve the memory issue at scale in a regional NWP. Different from these works, we focus on the algorithm improvements for solving the linear systems in global model of GRAPES in this paper.

3. OPTIMIZATION STRATEGY

For Equation (6), there is at most 19 nonzeros B_i in each row of the coefficient matrix [3, 18, 19]. We can form a sparse asymmetric linear system in Equation (7) by coordinate ordering all of these equations.

$$Ax = b \tag{7}$$

Figure 3 shows the sparse pattern of the matrix associated with $4 \times 4 \times 4$ grid points.

We take GRAPES global model with 100-km horizontal resolution and 36 vertical levels as an example. It has $360 \times 180 \times 36$ grid points in total. Table I lists the maximum and minimum eigenvalues and condition number of matrix A.

The aforementioned table shows that the condition number of matrix A is about 30,000. The convergence rate of the iterative algorithm is slower when the condition number is so large. So, we need to carefully design the preconditioning to speed up the convergence of iterative algorithm.

3.1. Improvement on iterative algorithm

Iterative methods are commonly used for solving large-scale Helmholtz equations. Krylov subspace method is an important kind of iterative method, for example, GCR, GMRES, BiCGSTAB, and IDR. GCR is the generalized conjugate gradient method for nonsymmetric problems, which is easy to implement. And GMRES is one of the most common algorithms, which is mathematically equivalent to GCR but avoids the issue of breakdown. BiCGSTAB is based on CGS and solves the irregular convergence problem of CGS [20]. Recently, IDR algorithm is widely used because of its

Figure 3. The sparse pattern of the matrix associated with $4 \times 4 \times 4$ grid points.

Table I. The numerical characteristics of matrix A.

Maximum eigenvalue	Minimum eigenvalue	Condition number
1.9999	6.738e-5	29680.9

DOI: 10.1002/cpe

Table II. The average computing cost of one iteration for different iterative methods.

Method	Inner product	SAXPY	Matrix–vector product	Precond solve
GMRES(k)	$\frac{k+3}{2}$	$\frac{k+3}{2}$	1	1
GCR(k)	$\frac{k+3}{2}$	$\frac{k+3}{2}$	2	1
BiCGSTAB	4	6	2	2
IDR(k)	k+2	$\frac{5k^2+11k+2}{2k}$	$\frac{k+1}{k}$	$\frac{k+1}{k}$

SAXPY, Scalar Alpha X Plus Y; GMRES, generalized minimal residual method; GCR, generalized conjugate residual method; BiCGSTAB, biconjugate gradient stabilized method; IDR, induced dimension reduction method.

Table III. Iterations with different algorithms for GRAPES.

Algorithm	GCR	GMRES	BiCGSTAB	IDR(1)	IDR(2)	IDR(5)
Iterations	253	650	363	374	426	474

GRAPES, Global/Regional Assimilation and Prediction System; GCR, generalized conjugate residual method; GMRES, generalized minimal residual method; BiCGSTAB, biconjugate gradient stabilized method; IDR, induced dimension reduction method.

fewer global communication times. Table II lists the computational costs of one iteration on average for different iterative algorithms, where k is the restart number. Table III shows the number of iterations with different algorithms for test case with $360 \times 180 \times 36$ grid points in GRAPES global model. No preconditioning is used in these tests, and the convergence threshold is 1e-6 for 2-norm of the residual vector.

According to Tables II and III, the GCR algorithm has a better performance than GMRES with the same restart number of 10 because it only needs only about one-third iterations of GMRES. The IDR algorithm is proved to be suitable for the strongly asymmetrical matrices, while it does not fit to the matrix of GRAPES so well. The obvious numerical perturbation is observed in our tests using IDR methods, which can be found in detail in Figure 4. The computational costs between GCR and BiCGSTAB are comparable as well as the iteration number in Table III. Therefore, using these two methods, we further test the first time step of GRAPES global model with $360 \times 180 \times 36$ grid points with blocked ILU(0) preconditioning over different processing cores. The restart number of GCR we used is 10. In Figure 5, we find that GCR method needs over 15% fewer iterations on average than the BiCGSTAB method for GRAPES, which leads to obtaining a better performance. According to the comparisons, we choose GCR algorithm as the base of our further work.

The restart GCR algorithm is described as follows, where k is the restart number. In the GCR algorithms, each iteration needs two global communications (steps 3 and 8), which cannot be combined because of data dependency. When the process number increased, the communication overhead becomes the bottleneck of the whole computation, accounting for about 60% of the solving time, causing the pool scalability of the GCR algorithm (Figure 6).

On the basis of the classical GCR method, we define the vectors $q_n = Ap_n$,$t_j = AR_j$. Then, we can obtain the following forms for subspace vectors:

$$(R_{j+1}, Ap_{j+1}) = (R_{j+1}, AR_{j+1}) + \sum_{i=0}^{j} \beta_{ij}(R_{j+1}, Ap_i) = (R_{j+1}, t_{j+1})$$

$$(q_{j+1}, q_{j+1}) = (t_{j+1}, t_{j+1}) + 2\sum_{i=0}^{j} \beta_{ij}(t_{j+1}, q_i) + \sum_{i=0}^{j} \beta_{ij}^2(q_i, q_i) = (t_{j+1}, t_{j+1}) - \sum_{i=0}^{j} \beta_{ij}^2(q_i, q_i)$$

On the basis of the two equations earlier, the IGCR algorithm can be depicted in Algorithm 2. Note that the inner products of steps 8, 9, and 11 are independent and they can be combined as one

DOI: 10.1002/cpe

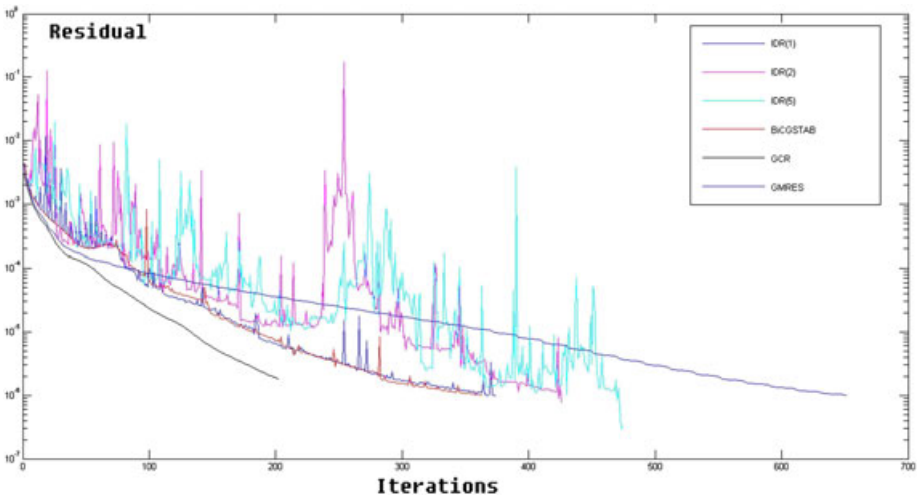

Figure 4. Residuals of different algorithms without preconditioning.

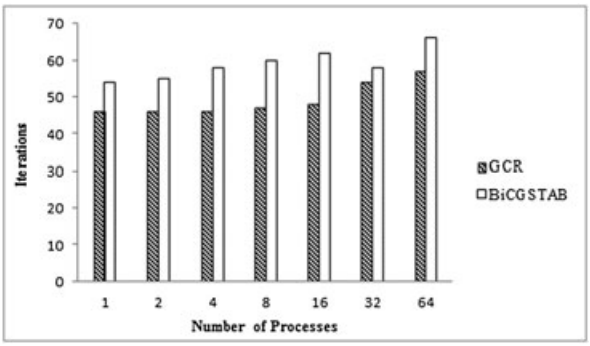

Figure 5. The number of iterations by generalized conjugate residual (GCR) and biconjugate gradient stabilized method (BiCGSTAB).

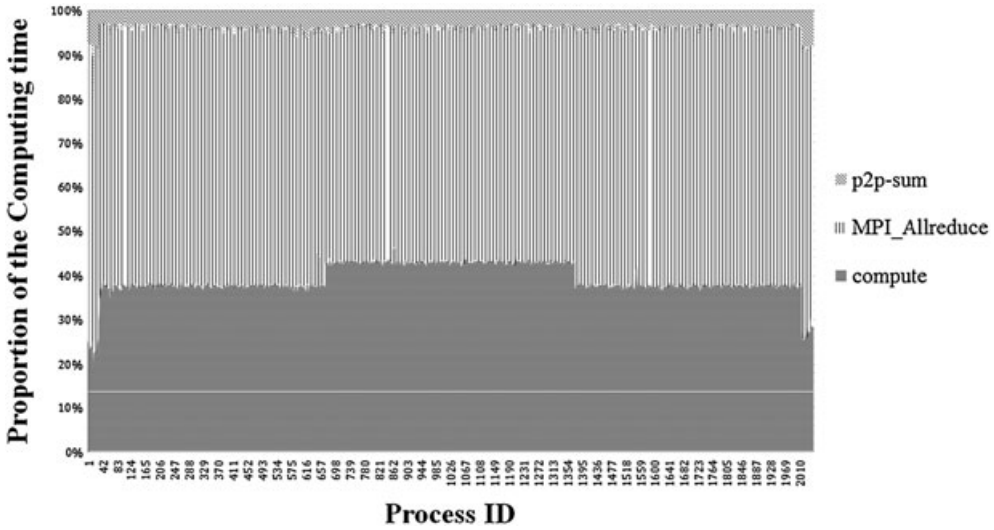

Figure 6. Analysis for timing of Helmholtz solver in the 2048 processes.

DOI: 10.1002/cpe

Algorithm 1 The GCR(k) method

01. *Compute* $R_0 = b - Ax_0$, $R'_0 = M^{-1}R_0$, $p_0 = R'_0$
02. *Do i=1,2,...,until convergence*
03. $\alpha_{i-1} = (R_{i-1}, Ap_{i-1})/(Ap_{i-1}, Ap_{i-1})$
04. $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
05. $R_i = R_{i-1} - \alpha_{i-1} q_{i-1}$
06. $R'_i = M^{-1} R_i$
07. *Do j=int[(i-1)/k],...,i-1*
08. $\beta_{ij} = -(AR'_i, Ap_j)/(Ap_j, Ap_j)$
09. *EndDo*
10. $p_i = R'_i + \sum_{j=int[\frac{i-1}{k}]k}^{i-1} \beta_{ij} p_j$
11. *EndDo*

Algorithm 2 The improved GCR method

01. *Compute* $R_0 = b - Ax_0$, $R'_0 = M^{-1}R_0$, $p_0 = R'_0, q_0 = Ap_0, \alpha_0 = (R_0, q_0), \gamma_0 = (q_0, q_0)$
02. *Do i=1,2,...,until convergence*
03. $\alpha_{i-1} = \alpha_{i-1}/\gamma_{i-1}$
04. $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
05. $R_i = R_{i-1} - \alpha_{i-1} q_{i-1}$
06. $R'_i = M^{-1} R_i$
07. $ar = AR'_i$
08. $\alpha_i = (R_i, ar)$
09. $c_i = (ar, ar)$
10. *Do j=int[(i-1)/k],...,i-1*
11. $\beta_{ij} = -(ar, q_j)/(q_j, q_j)$
12. *EndDo*
13. $\gamma_i = c_i - \sum_{j=int[\frac{i-1}{k}]k}^{i-1} \beta_{ij}^2 \gamma_j$
14. $p_i = R'_i + \sum_{j=int[\frac{i-1}{k}]k}^{i-1} \beta_{ij} p_j$
15. $if(i\%k == 0)\ q_i = Ap_i$
16. $else\ q_i = ar + \sum_{j=int[\frac{i-1}{k}]k}^{i-1} \beta_{ij} q_j$
17. *EndDo*

global communication in one iteration. Also, we extend the work in [13] to a restart version of preconditioned GCR algorithm in Algorithm 2. Although the proof on rigorous equivalence between two algorithms is still under investigation, the convergence rate of our improved GCR method is almost the same as the previous GCR method in all of our tests.

From the viewpoint of computation cost, we only add several vector updates for calculating γ_i and q_i, which do not require any communication and can be carried out with high-performance Basic Linear Algebra Subprograms. Therefore, the cost of IGCR algorithm on distributed memory computers can be significantly reduced because of fewer global communications. From the test results of Figures 7 and 8, the computing time of the IGCR algorithm is only slightly less than that of the original GCR algorithm when the number of processes is relatively small such as 256 and 512 cores. This is because the reduction of communication overhead of the IGCR algorithm is comparable with the increase for more computations using relatively low parallelism. However, the performance improvements are over 20% when the number of processes increases to 1024 and 2048 cores because the timing for global communication becomes the major proportion of the total time. It is a little bit strange that the performance improvement over 1024 cores is better than that

DOI: 10.1002/cpe

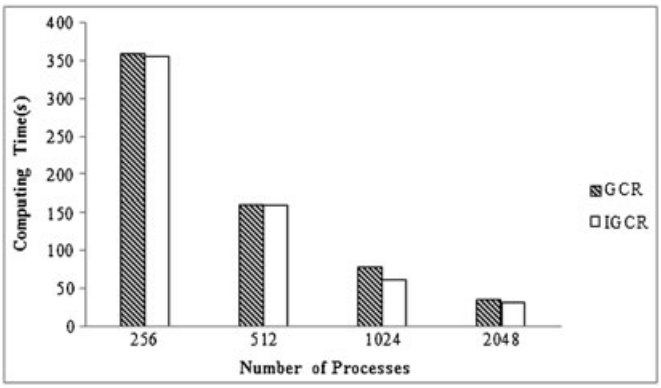

Figure 7. The measured computing time with different processing cores between the generalized conjugate residual (GCR) method and improved GCR (IGCR) algorithms.

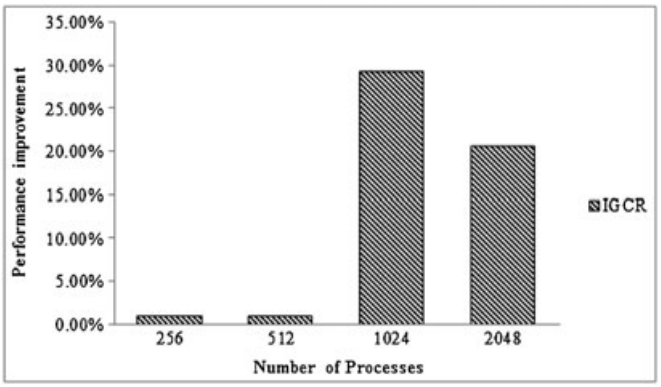

Figure 8. The performance improvement with different processing cores for the improved generalized conjugate residual (IGCR) algorithm to the GCR algorithm.

over 2048 cores in Figure 8. It might be due to the irregular topology mapping over the Tianhe-1A system, which increases the communication overhead over 1024 cores. Note that the test cases have $2400 \times 1200 \times 36$ grid points in total and are running for 6-h weather prediction using GRAPES global model. And the time measured in Figure 8 is only the time for solving Helmholtz equations. The convergence threshold is 1e-6 for the 2-norm of residual vector in the tests.

3.2. Optimization on preconditioner

The selection of different preconditioners has a great impact on the convergence performance of iterative algorithms. Among known preconditioning techniques, ILU is the most promising one. We compare the GCR algorithm with several blocked ILU-k preconditioners. Figure 9 shows the number of iterations with different degrees of parallelism and different levels of ILU preconditioning. It is easy to understand that the more level k used, the less iterations are paid for solution. In Figure 9, we use the test cases having $1440 \times 720 \times 36$ grid points in total.

For GRAPES, after scaling with diagonal elements, the values of the coefficient matrix have the following properties with the same test cases in Figure 9:

$$B1 = 1.0, B10/B15 \sim 1e^{-1}, B2/B3 \sim 1e^{-2}, B4/B5 \sim 1e^{-3}, \text{and all others} \leqslant 1e^{-5}.$$

So, we use the largest 7-diagonal matrix and the full matrix to form the preconditioned matrices by ILU(0) and compare the convergence rate of the two preconditionings in Table IV. After 7-diagonal and 19-diagonal preconditionings, the condition numbers of matrix A changed to 1121.9 and 1059.5, which is very close. Although preconditioning using the 19-diagonal matrix has 10%

DOI: 10.1002/cpe

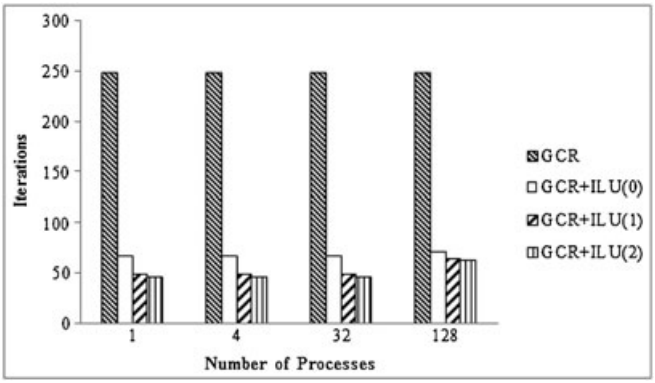

Figure 9. Iterations of generalized conjugate residual method (GCR) with incomplete LU factorization with level k fill (ILU(k)).

Table IV. Convergence and matrix characteristics after ILU(0) preconditioning.

Method	Iterations	Maximum eigenvalue	Minimum eigenvalue
19-diagonal ILU(0)	59	1.9072	0.0018
7-diagonal ILU(0)	66	1.9073	0.0017

ILU, incomplete LU factorization.

fewer iterations, the computation cost of 7-diagonal preconditioner is much less than that of the 19-diagonal matrix. So, the computing time of preconditioning over the 19-diagonal matrix will become longer than that over 7-diagonal matrix.

To reduce the iteration number, ILU(2) preconditioner is adopted instead of ILU(0) in our work. Although ILU(2) has more computations than ILU(0) in one iteration, it can improve the convergence rate of calculation and can further reduce the overhead for communication. When the degree of parallelism is large, the overhead of global communication will sharply increase. By reducing the iteration number, we can decrease the number of global communication as well as the computing time. At the same time, we use the 7-diagonal matrix for preconditioning to release the computation burden introduced by using ILU(2) on the full matrix for preconditioning, which tries to enhance the performance further.

The ILU(2) preconditioning we use is a kind of ILU(p) algorithm. The process of ILU(p) algorithm is shown as follows [9].

Algorithm 3 ILU(p) Algorithm

01. *For all nonzero elements* a_{ij}*,define* $lev(a_{ij}) = 0$
02. *For i=2,...,n, Do*
03. *For each k=1,...,i-1 and for* $lev(a_{ik}) \leqslant p$ *Do*
04. *Compute* $a_{ik} = a_{ik}/a_{kk}$
05. *Compute* $a_{i*} = a_{i*} - a_{ik}a_{k*}$
06. *Update the levels of fill of the nonzero* $a_{i,j}s$ *using* (8)
07. *EndDo*
08. *Replace any element in row i with* $lev(a_{ij}) > p$ *with zero*
09. *EndDo*

In which, the initial level of an element a_{ij} is defined by

$$\text{lev}_{ij} = \begin{cases} 0 & a_{ij} \neq 0 \; or \; i = j \\ \infty & \text{otherwise} \end{cases}$$

DOI: 10.1002/cpe

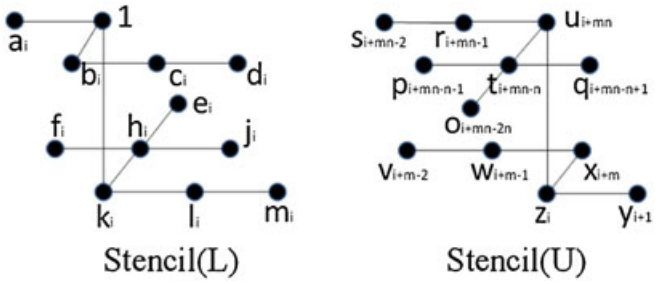

Figure 10. Stencil associated with the product of the L and U factors in the incomplete lower and upper 2 (ILU(2)) factorization for the largest 7-diagonal matrix.

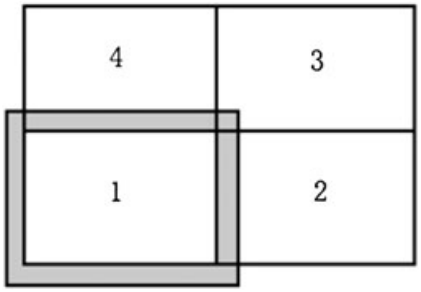

Figure 11. Domain decompositions of Global/Regional Assimilation and Prediction System model.

and

$$\mathrm{lev}_{ij} = \min\left\{\mathrm{lev}_{ij}, \mathrm{lev}_{ik} + \mathrm{lev}_{kj} + 1\right\} \tag{8}$$

We can use $\mathrm{stencil}_i(A)$ to represent the stencil of the matrix A at the grid point labeled i (Figure 10).

Compared with ILU(0) preconditioner, ILU(2) has more fill-in elements.The form of computing $\mathrm{stencil}_i(LU)$ is as follows:

$$\begin{aligned}\mathrm{stencil}_i(LU) &= 1 \times \mathrm{stencil}_i(U) + a_i \times \mathrm{stencil}_{i-1}(U) + b_i \times \mathrm{stencil}_{i-m}(U)\\ &+ c_i \times \mathrm{stencil}_{i-m+1}(U) + d_i \times \mathrm{stencil}_{i-m+2}(U) + e_i \times \mathrm{stencil}_{i-mn+2m}(U)\\ &+ f_i \times \mathrm{stencil}_{i-mn+m+1}(U) + h_i \times \mathrm{stencil}_{i-mn+m}(U) + j_i \times \mathrm{stencil}_{i-mn+m+1}(U)\\ &+ k_i \times \mathrm{stencil}_{i-mn}(U) + l_i \times \mathrm{stencil}_{i-mn+1}(U) + m_i \times \mathrm{stencil}_{i-mn+2}(U)\end{aligned} \tag{9}$$

3.3. Optimization on preconditioning over domain decomposition

Solving the Helmholtz equation uses two-dimensional mesh partition, which is the same as the domain decomposition in the GRAPES model. We use four processes as an example shown in Figure 11. The grid is divided by four processes, and each process has its local region. The white areas refer to the computing area for the corresponding processes, and the gray part is the overlapping (halo) area of process 1. The results in the overlapping area need to exchange with neighbor processes during iterations.

The physical blocked Jacobi preconditioner is adopted in the original version of GRAPES global model, which is to calculate the ILU-k preconditioner in the local area (the white area 1 in Figure 11) only. For the overlapping (halo) region (the gray are a in Figure 11), data are set to 0. The benefit of this method is that it does not require any communication while do preconditioning. The disadvantage of this method is that it ignores the relationship among domains, which leads to slow convergence, especially using large number of processes.

To improve the convergence of the GCR algorithm, we use the RAS method together with the ILU-k preconditioning. For example, consider a two-dimensional mesh partition in Figure 12.

DOI: 10.1002/cpe

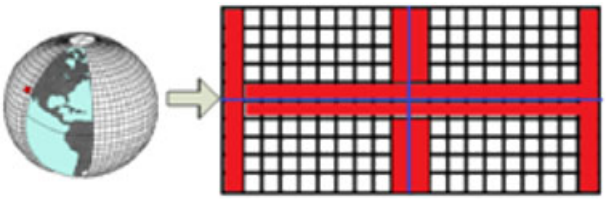

Figure 12. Restricted additive Schwarz method for Global/Regional Assimilation and Prediction System model.

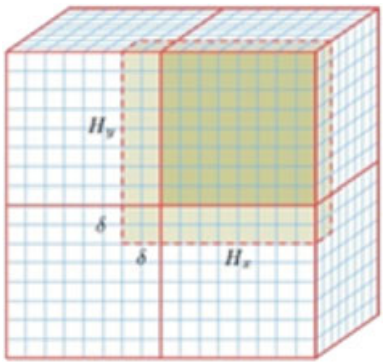

Figure 13. Domain decompositions with overlaps [21].

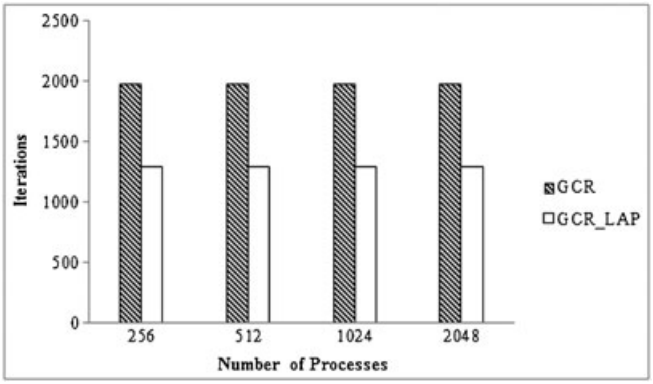

Figure 14. The comparison between our proposed preconditioning and the original one. GCR, generalized conjugate residual method.

The red parts refer to the overlapping regions between partitions. The proposed preconditioner uses not only the local data to update the vectors but also the data of overlapping areas from neighbor processes during ILU(2) preconditioning.

In Figure 13, the red solid lines show the partition of the domain, which has $4 = 2 \times 2$ nonoverlapping subdomains of size $H_x \times H_y \times H_z$. H_x refers to the length along with latitude, and H_y refers to the length along with longitude in one domain. The dotted lines show the extend boundaries of an overlapping subdomain. The value of δ in this paper is set to 1. For each overlapping subdomain, whose size is $(H_x + \delta) \times (H_y + \delta) \times H_z)$, we use ILU(2) as the local preconditioner.

Both the domain decomposition method we proposed in this section and the local preconditioner in the previous section try to improve the convergence by increasing the computation cost and the neighboring communication operation for iteration. Our tests prove that it is worth performing these optimizations on preconditioning. For 6-h prediction using GRAPES global model with $1440 \times 720 \times 36$ grid points, the convergence results of the original algorithm and our proposed

DOI: 10.1002/cpe

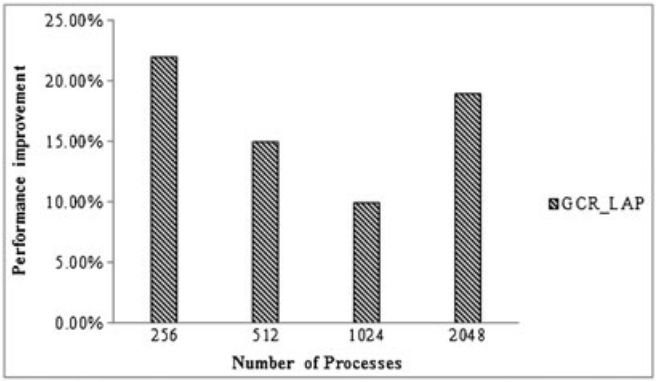

Figure 15. The performance improvement for IGCR_LAP algorithms and the original one. IGCR, improved generalized conjugate residual method.

Table V. The configuration of the Tianhe-1A System.

Item	Parameter
CPU	2*Intel Xeon X5670/node
Clock Frequency	2.93GHz
Memory	36G/node
OS	RedHat Enterprise Linux 5.3 x86_64
Network	YH-NET
Filesystem	YH-lustre

algorithm (GCR_LAP) are shown in Figure 14. The GCR_LAP algorithm can decrease the total iterations by over 30%.

Furthermore, the speedups of GCR_LAP compared with the original algorithm have also been shown in Figure 15. The performance improvement can achieve from 10% to 22% using 256 to 2048 cores. Our proposed preconditioning algorithm is to form a more accurate preconditioner to reduce the number of iterations. Although increasing the computation cost as well as the communication overhead in each iteration, the new preconditioning across domains can improve performance since it reduces the number of iterations.

4. PERFORMANCE EVALUATION

This work is carried out on the Tianhe-1A system at the National Supercomputer Center in Tianjin, whose configuration is shown in Table V.

The test case is 0.15 degree horizontal resolution, 36 vertical levels in GRAPES global model with space discretization of $2400 \times 1200 \times 36$ grid points. We perform a 6-h simulation with the initial and forcing conditions by the data assimilation component of GRAPES. The performance for the Helmholtz equations are measured and compared for the original used preconditioned GCR method (GCR) and the improved GCR with our proposed preconditioning (IGCR_LAP). The parallel degrees tested are 256, 512, 1024, and 2048.

In Figure 16, we can find that the time for Helmholtz equations of the 6-h simulation with GRAPES can shorten to no more than 30 s with 2048 cores. That is to say, we can complete the solution of Helmholtz equations for 10-day prediction in 20 min. As shown in Figure 17, the IGCR_LAP method we proposed decreases the time consumed for Helmholtz equations by 18% to 40% using 256 to 2048 cores, comparing with the original GCR method using ILU(0) and block Jacobi preconditioning. The reason why we can obtain a better performance is that the proposed algorithm not only reduces the iteration number but also cuts down half of the global communications, which benefits all of the parallel degrees. The observed performance improvement over 1024 cores is better than

DOI: 10.1002/cpe

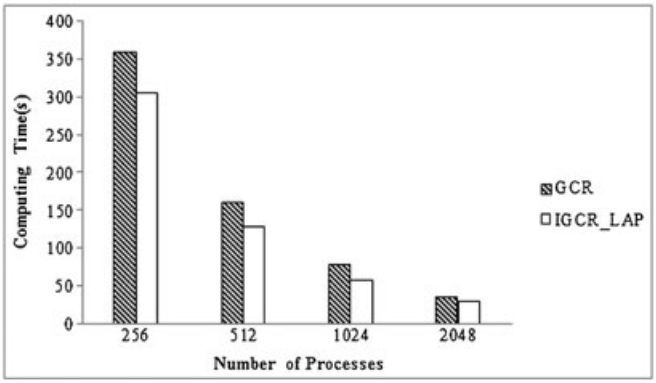

Figure 16. The measured computing time with different processing cores for the generalized conjugate residual (GCR) and improved GCR (IGCR)_LAP algorithms.

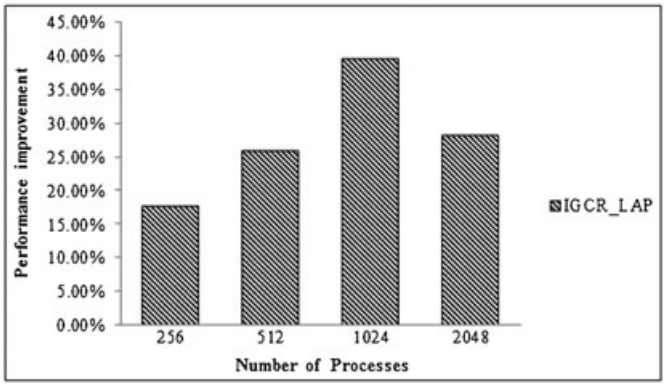

Figure 17. The performance improvement with different processing cores for the improved generalized conjugate residual (IGCR)_LAP algorithm.

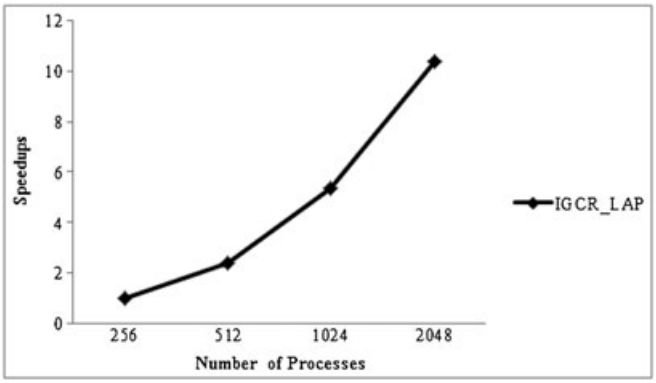

Figure 18. Speedups with different processing cores for the improved generalized conjugate residual (IGCR)_LAP algorithms.

that over other cores might be due to the irregular topology mapping over the Tianhe-1A system, which is sharing by a large number of users. Moreover, the scalability of the IGCR_LAP algorithm can obtain a very good speedups compared with the performance result of 256 cores (Figure 18). Also, the cache effect makes a contribution to the superlinear speedups shown in Figure 18.

DOI: 10.1002/cpe

5. CONCLUSION

Our work in this paper focuses on the performance optimization on the Helmholtz equation, which is the most computation intensive part in the GRAPES global model. After analyzing the numerical characteristics and the convergence of different iterative algorithms, we choose GCR as the baseline algorithm for use and further propose an advanced preconditioner and an improved restarted GCR algorithm for large-scale sparse linear system in GRAPES over scale. The proposed preconditioner combines high-level ILU-k preconditioner over 7-diagonals of the coefficient matrix with the RAS method, and the IGCR algorithm reduces the global communication operations by refactoring the restarted GCR algorithm. Performance evaluation on the Tianhe-1A system shows that the new preconditioning technique reduces almost one-third iterations for solving the linear systems, and the proposed methods can obtain 25% performance improvement on average compared with the original version of Helmholtz solver in GRAPES. The speedup with our algorithms can reach 10 using 2048 cores compared with 256 cores on the Tianhe-1A system.

ACKNOWLEDGEMENTS

The authors would like to thank Prof. Haixiang Lin, Dr. Ke Wang, and Dr. Shiming XU for the extended discussion on scalable algorithms for linear systems and the National Supercomputer Center in Tianjin for providing access to the Tianhe-1A system. This work is supported in part by the 973-Program of China under grant 2010CB951903, the 863-Program of China under grant 2010AA012302 the NSF China under grants 51190101 and 60973143, the Scientific and Technical Supporting Programs under grant 2009BAH42B0202, and the Tsinghua National Laboratory for Information Science and Technology.

REFERENCES

1. Chen DH, Xue JS, Hu JL. A GRAPES new dynamical model and its preliminary numerical tests. *Proceedings of International Workshop on NWP Models for Heavy Precipitation in Asia and Pacific Areas, Japan Meteorological Agency and Ship & Ocean Foundation,* Tokyo, Japan 2003:142–146.
2. Liu Y, Cao JW. ILU preconditioner for NWP system: GRAPES. *Computer Engineering and Design* 2008; **29**(3):731–734.
3. Xue JS, Chen DH. *Scientific Design and Application of the Numerical Prediction System GRAPES*, Chapter 2. Science Press: Beijing, China, 2008.
4. Zhang LL, Gong XP, Song JQ. Parallel preconditioned GMRES solvers for 3-D Helmholtz equations in regional non-hydrostatic atmosphere model. *International Conference on Computer Science and Software Engineering* 2008; **3**:287–290.
5. Zhang LL. Research of high-performance parallel computing for numerical models in meteorologic prediction. *PhD thesis*, National University of Defense Technology, 2002, Hunan,China.
6. Liu GP, Zhao WT, Zhang LL. Parallel Helmholtz solvers for Chinese GRAPES atmosphere model based on PETSc tools. *Proceedings of International Symposium on Distributed Computing and Applications to Business, Engineering and Science*, Hubei, China, 2007; 287–290.
7. Benzi M. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics* 2002; **182**(2):418–477.
8. Bader D. *Petascale Computing: Algorithms and Applications, Simulating Cosmological Evolution with Enzo*. CRC Press: Boca Raton, USA, 2007.
9. Saad Y. *Iterative Methods for Sparse Linear Systems*, 2nd ed. SIAM: Philadelphia, USA, 2003.
10. Briggs L, Henson V, McCormick F. *A Multigrid Tutorial*. SIAM: Philadelphia, USA, 2000.
11. Cai XC, Sarkis M. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing* 1999; **21**(2):792–797.
12. Cai XC, Dryja M, Sarkis M. A convergence theory for restricted additive Schwarz methods. *Technical Report*, Dept. of Computer Science, Univ. of Colorado at Boulder, 1999.
13. Zhao LB, Tian YX. Improved parallel generalized conjugate residual algorithm. *Computer Engineering* 2009; **35**(4):80–82.
14. Yang L, Brent R. The improved BiCGSTAB method for large and sparse unsymmetric linear systems on parallel distributed memory architectures, Beijing, China, 2002; 324–328.
15. Sonneveld P, Gijzen V. IDR (s): a family of simple and fast algorithms for solving large nonsymmetric linear systems. *SIAM Journal on Scientific Computing* 2008; **31**(2):1035–1062.
16. Daniel B, Henry J. Experiences in optimizing a numerical weather prediction model: an exercise in futility? *7th Linux Cluster Institute Conference*, Norman, USA, 2006; 1–34.

DOI: 10.1002/cpe

17. Alvaro L, Jairo P, Daniel M. Challenges and solutions to improve the scalability of an operational regional meteorological forecasting model. *International Journal of High Performance Systems Architecture* 2011; **3**(2–3):87–97.
18. Yang XS, Hu JL, Chen DH. Verification of GRAPES unified global and regional numerical weather prediction model dynamic core. *Chinese Science Bulletin* 2008; **53**(22):3458–3464.
19. Chen DH, Xue JS, Yang XS. New generation of multi-scale NWP system (GRAPES): general scientific design. *Chinese Science Bulletin* 2008; **53**(22):3433–3455.
20. Barrett R, Berry M, Chan T. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, Chapter 3. SIAM: Philadelphia, USA, 1994.
21. Yang C, Cai XC. Parallel multilevel methods for implicit solution of shallow water equations with non-smooth topography on the cubed-sphere. *Journal of Computational Physics* 2011; **230**(7):2523–2539.

DOI: 10.1002/cpe